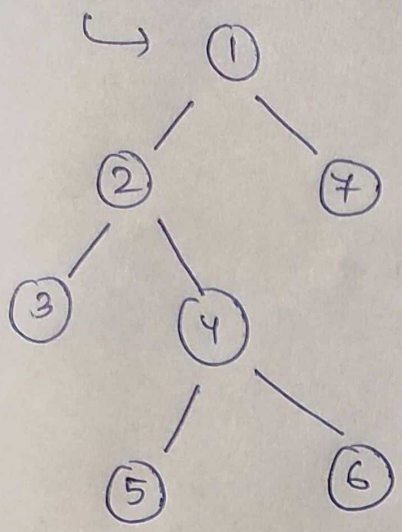
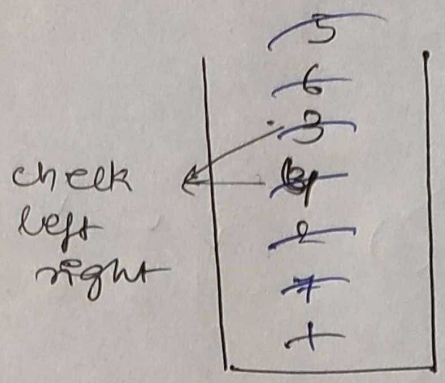
Iterative Preorder:- (Root left Right)





Stack → becomes empty.

We need to traverse through left then right but stack is LIFO data structure, so we will push right element then the left element in ds.

1 2 3 4 5 6 7

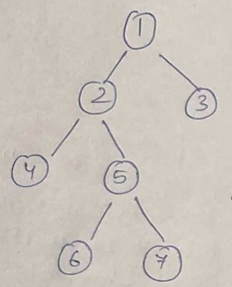
Code :-

```
vector<int> preorderTraversal (TreeNode * root) {
        vector<int> preorder;
        if(root == NULL) return preorder;

        stack< TreeNode *> st;
        st.push(root);
        while(! st.empty()) {
            root = st.top();
            st.pop();
            preorder.push_back (root->val);
              if (root->right ! = NULL) {
                    st.push(root->right);
                }
              if (root-> left ! = NULL) {
                    st. push (root-> left);
                }
            }
        ret
```

Iterative Traversal of (Inorder)

↳ Left Root Right.



4 2 6 5 7 1 3

node = 1 (left)

2

4

~~null~~ (As soon as left becomes null put then in stack and print it if (left/right) becomes null.

~~null~~ & pop from stack.

5

6

~~null~~ print & pop from

~~null~~ print stack.

7

~~null~~

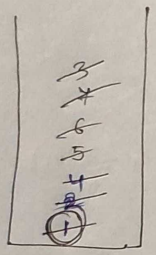
~~null~~

(1) left in stack pop & print.

~~3~~

~~null~~

~~null~~



Stack

↳ empty

Code :-

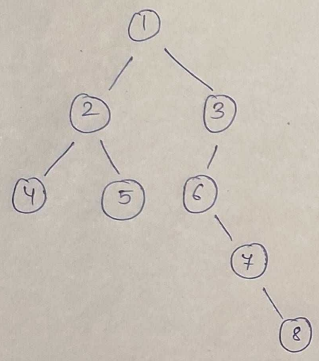
```
vector<int> inOrderTraversal (TreeNode *root) {
    stack <TreeNode *> st;
    TreeNode * node = root;
    vector<int> inorder;
    while(true) {
        if(node != NULL) {
            st.push(node);
            node = node->left;
        }
        else {
            if(st.empty() == true) break;
            node = st.top();
            st.pop();
            inorder.push_back(node->val);
            node = node->right;
        }
    }
    return inorder;
}
```

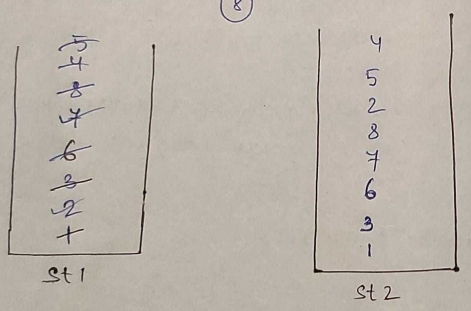
TC : O(n)

SC : O(n) ≈ O (Height of Binary Tree)

Iterative PostOrder Traversal.

(Left Right Root) → Using 2 stack.





If root(1) has left and right take and put them in stack

Put st.top (now 3) in stack-2.

Now st→left(6)

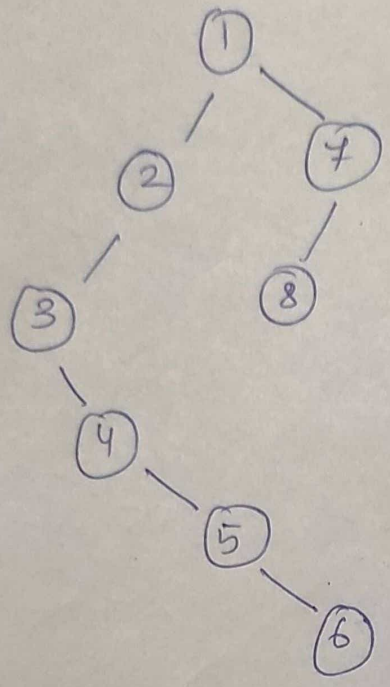
As soon as st1 is empty take all element out from st2 in LIFO order.

```
vector<int> postOrderTraversal (TreeNode *root) {
    vector<int> postorder;
    if (root == NULL) return postorder;
    stack< TreeNode *> st1, st2;
    st1.push(root);
    while (!st1.empty()) {
        root= st1.top();
        st1.pop();
        st2.push(root);
        if(root->left != NULL) {
            st1.push(root->left);
        }
        if(root->right!= NULL) {
            st1.push(root->right);
        }
    }
    while (!st2.empty()) {
        postorder.push_back(st2.top()->val);
        st2.pop();
    }
    return postorder;
}
```

TC: O(n) SC: O(n+n)

: O(2n)

Iterative Postorder
↳ using 1 stack.



6 5 4 3 2 8 7 1

temp = ~~4~~ ~~5~~ ~~6~~ null
~~6~~ ~~5~~ ~~4~~ ~~3~~ null
~~2~~ ~~1~~ null ~~8~~ null
~~1~~

curr = ~~1~~ ~~2~~ ~~3~~ null
~~4~~ null
~~5~~ null
~~6~~ null
~~7~~ ~~8~~
null

Code:-

```
while (curr != null || !st.empty()) {
        if (curr != null)
            st.push(curr);
            curr = curr->left;
        else
            temp = st.top()->right;
            if (temp == null)
                    temp = st.top();
                    st.pop();
                    post.push_back(temp);
    while (!st.empty() && temp == st.top()->right)
        temp = st.top(), st.pop();
        post.add(temp->val);
    else
        curr = temp;
}
```